

JDBC的使用

使用JDBC存取数据库

使用JDBC来存取数据库通常包含以下几个步骤：

- 载入JDBC driver
- 在客户程序与数据库之间建立连接
- 通过数据库连接将SQL语句从Java程序传到数据库
- 通过数据库返回的记录集得到所需的数据
- 如需要，再用修改后的数据更新数据库
- 操作结束，关闭连接

前面讲过，JDBC API作为java核心类库的一部分，直接包含在JDK软件包中， 其对应的包为`java.sql`，要使用JDBC，必须在程序开始import这个包：`import java.sql.*`;

以下分别来说明使用JDBC来存取数据库的几个步骤。

一、载入JDBC driver

- 使用类型1的driver(JDBC-ODBC bridge driver)：类型1的driver直接包含在JDBC软件中，只要在客户端安装好数据库对应的ODBC driver即可使用。程序中可使用下列语句载入类型1的driver：

`Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");`

- 使用类型2和类型4的JDBC driver：针对不同类型的数据库，首先要从厂家那里获取相应数据库的JDBC driver，再按说明将它载入到程序中。下面以oracle为例说明：先从oracle的网站下载指定数据库版本的JDBC driver，然后将该driver的存放路径加入CLASSPATH，最后在程序中使用以下语句载入driver：

`Class.forName("oracle.jdbc.driver.OracleDriver");`

二、与数据库建立连接

使用JDBC DriverManager类的getConnection()方法与指定的数据库建立连接。如：

```
Connection con =  
DriverManager.getConnection(url,db_username, db_password);
```

其中的db_username和db_password分别对应所连数据库的用户名和口令。url与网络中的URL有所不同，它给出了要连数据库的有关信息。**不同类型的JDBC driver对应不同格式的url**。即使是同一类型的JDBC driver，不同厂家的数据库其格式也有所不同。例如：

- ❶ 类型1的driver的url的格式

jdbc:odbc:dsn_name

其中，dsn_name为数据库对应的ODBC driver的DSN名

- ② 类型4的Oracle driver (the thin driver) 的格式:

`jdbc:oracle:thin@database_name:port_no:SID`

其中, `database_name`为数据库服务器的名字或IP地址, `port_no`为数据库listener的端口号, Oracle缺省为1521, `SID`为数据库的SID。如:

String url =

`"jdbc:oracle:thin:@192.168.0.150:1521:ora816";`

- ③ 类型2的Oracle driver (the OCI driver) 的格式:

`jdbc:oracle:oci8@database_name`

该`database_name`或者是SQL*Net中的name-value对 (pair), 或者,如果你使用Oracle Name server, 也可以使用`tnsname.ora`文件中的name。

三、建立Statement对象

使用Connection对象的createStatement()方法建立一个Statement对象:

```
Statement stmt = conn.createStatement ();
```

四、执行SQL查询语句

一旦建立了Statement对象后，就可以利用该对象的executeQuery()方法让数据库执行指定的SQL语句，SQL语句是作为方法的参数传入并递交数据库去执行，执行后的结果放在一个ResultSet类型的对象中作为方法的返回值。如：

```
ResultSet rset = stmt.executeQuery ("SELECT ename from  
emp where empno = 7900");
```

五、获取数据库查询结果

调用ResultSet对象的next()方法得到该集合中新的一行。如果ResultSet集合多于一行，可以用一个循环将它取出。如：

```
boolean more=rset.next();
while (more) {
    String ename = rset.getString("ename")
    System.out.println(ename);
    more=rset.next();
}
```

其中，ResultSet的getXXX()方法用于获取该行中指定列的值。如对应的SQL类型为字符串或文字类型，可使用getString()方法。如对应的SQL类型为数值类型，如INTEGER, FLOAT, NUMBER等，则可使用getInt(), getFloat()等方法。另外，getXXX()方法的参数也可以是列号，假如上述ename为ResultSet中的第一列，则getString(1)与getString("ename")效果一样。

六、更新数据库

有时不仅要检索数据库中的数据，还需更新数据库的内容。对数据库的更新操作通常使用 `executeUpdate()` 方法，它通常用来执行 CREATE, INSERT, UPDATE 或 DELETE 等操作，该方法的返回值为 int 类型的数值，代表数据库中已更新的行数，如没什么可返回的，返回值为 0。

➤ 创建表：

```
String createTableCoffees = "CREATE TABLE COFFEES " +  
    "(COF_NAME VARCHAR(32), SUP_ID INTEGER, PRICE  
    FLOAT, " + "SALES INTEGER, TOTAL INTEGER)";  
stmt.executeUpdate(createTableCoffees);
```


- 在表中插入记录：

```
stmt.executeUpdate("INSERT INTO COFFEES " +  
"VALUES ('Colombian', 101, 7.99, 0, 0)");
```

- 更新记录：

```
String updateString = "UPDATE COFFEES " +  
"SET SALES = 75 " + "WHERE COF_NAME LIKE  
'Colombian';  
stmt.executeUpdate(updateString);
```

七、关闭数据库连接

对数据库操作结束后，通常要依次关闭打开的ResultSet, Statement和Connection对象,如：

```
rset.close();
```

```
stmt.close();
```

```
conn.close();
```

要注意的是，前面介绍的许多方法如createStatement(), executeQuery()和executeUpdate()等方法均会产生一个SQLException的例外，所以使用这些语句时要作相应的例外处理。

DriverManager

DriverManager用来建立和数据库的连接以及管理JDBC驱动程序，DriverManager的常用方法有：

方法	描述
registerDriver(Driver driver)	在DriverManager中注册JDBC驱动程序
getConnection(String url,String user,String pwd)	建立和数据库的连接，返回Connection对象
setLoginTimeOut(int seconds)	设定等待数据库连接的最长时间

Connection

Connection代表Java程序和数据库的连接，Connection的常用方法有：

方法	描述
getMetaData()	返回数据库的MetaData数据，MetaData数据中包含数据库的相关信息，例如当前数据库连接的用户名、使用的JDBC驱动程序等
createStatement()	建立并返回Statement对象
prepareStatement(String sql)	建立并返回PreparedStatement对象

Statement

Statement用来执行静态的SQL语句。如select、insert、Update、delete语句。Statement的常用方法有：

方法	描述
executeQuery(String sql)	用来执行静态的查询语句，返回结果是一个记录集
executeUpdate(String sql)	用来执行静态的更新语句，返回结果为影响的记录条数

例如：String sql = “select manager_name,manager_pwd
from d_manager where manager_id=1”;

ResultSet rs = stmt.executeQuery(sql);//stmt为Statement对象

ResultSet

ResultSet表示select语句查询得到的记录集和。ResultSet的常用方法有：

方法	描述
getString(int columnIndex) getString(String colomnName)	返回指定字段的String类型的指
getInt(int columnIndex) getInt(String colomnName)	返回指定字段的int类型的指
getFloat(int columnIndex) getFloat(String colomnName)	返回指定字段的float类型的指
next()	将记录指针向下移

编程示例

P207: 例1

P209: 例2

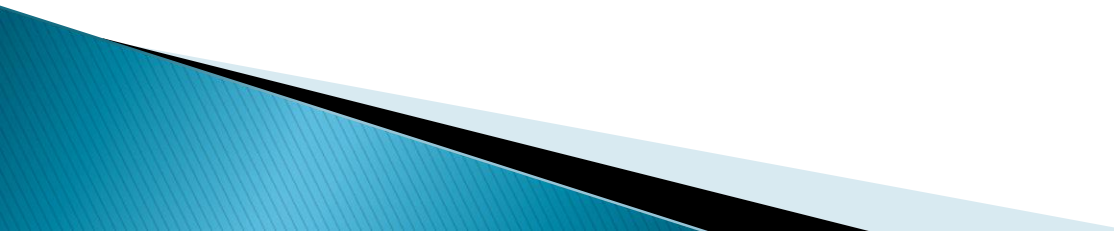
不同数据库JDBC连接方法

Oracle8/8i/9i数据库（thin模式）

```
Class.forName("oracle.jdbc.driver.OracleDriver").newInstance();  
String url="jdbc:oracle:thin:@localhost:1521:orcl"; //orcl为数据库的SID  
String user="test";  
String password="test";  
Connection conn= DriverManager.getConnection(url,user,password);
```

DB2数据库

```
Class.forName("com.ibm.db2.jdbc.app.DB2Driver ").newInstance();  
String url="jdbc:db2://localhost:5000/sample"; //sample为你的数据库名  
String user="admin";  
String password="";  
Connection conn= DriverManager.getConnection(url,user,password);
```



Sql Server7.0/2000数据库

```
Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver").newInstance();  
String url="jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=mydb";  
//mydb为数据库  
String user="sa";  
String password="";  
Connection conn= DriverManager.getConnection(url,user,password);
```

Sybase数据库

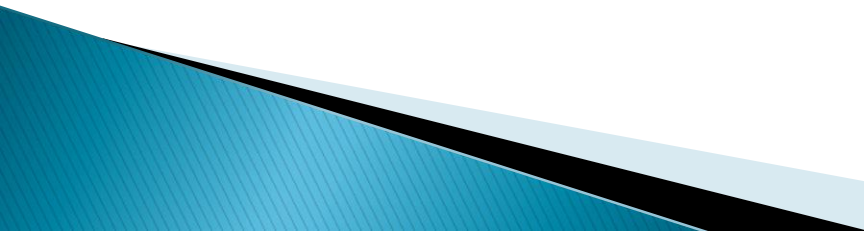
```
Class.forName("com.sybase.jdbc.SybDriver").newInstance();  
String url =" jdbc:sybase:Tds:localhost:5007/myDB";  
//myDB为你的数据库名  
Properties sysProps = System.getProperties();  
SysProps.put("user","userid");  
SysProps.put("password","user_password");  
Connection conn= DriverManager.getConnection(url, SysProps);
```

Informix数据库

```
Class.forName("com.informix.jdbc.IfxDriver").newInstance();  
String url = "jdbc:informix-sqli://123.45.67.89:1533/  
            myDB:INFORMIXSERVER=myserver;  
            user=testuser;password=testpassword"; //myDB为数据库名  
Connection conn= DriverManager.getConnection(url);
```

MySQL数据库

```
Class.forName("org.gjt.mm.mysql.Driver").newInstance();  
String url="jdbc:mysql://localhost/myDB?user=soft&password=soft1234  
            &useUnicode=true&characterEncoding=8859_1"  
//myDB为数据库名  
Connection conn= DriverManager.getConnection(url);
```



PostgreSQL数据库

```
Class.forName("org.postgresql.Driver").newInstance();  
String url = "jdbc:postgresql://localhost/myDB" //myDB为数据库名  
String user = "myuser";  
String password = "mypassword";  
Connection conn = DriverManager.getConnection(url, user, password);
```

access数据库直连用ODBC的

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver") ;  
String url = "jdbc:odbc:Driver={Microsoft Access Driver  
    (*.mdb)};DBQ="+application.getRealPath("/Data/ReportDemo.mdb");  
Connection conn = DriverManager.getConnection(url, "", "");  
Statement stmtNew = conn.createStatement() ;
```

事务处理

在Connection类中提供了3个控制事务的方法：

- **setAutoCommit**(boolean autoCommit):设置是否自动提交
- **commit**()：提交事务
- **rollback**()：撤销事务。例如：

```
try{ con = java.sql.DriverManager.getConnection(dburl,dbuser,dbpwd);
    con.setAutoCommit(false);
    stmt = con.createStatement();
    stmt.executeUpdate("update account set money=money-100 where name='zhangsan'");
    stmt.executeUpdate("update account set money=money+100 where name='lisi'");
    con.commit();
}catch(Exception ex){
    try{con.rollback();}
    catch(Exception e){e.printStackTrace();}
}
.....
```